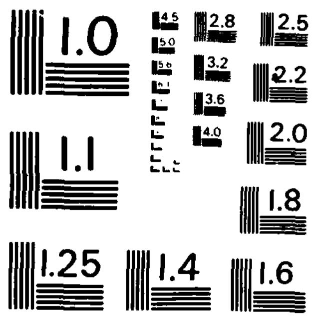
AD-A135 807 ANALYSIS OF RELIABLE MULTICAST ALGORITHMS FOR LOCAL NETWORKS(U) UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION S.. P V MOCKAPETRIS NOV 83 1/1

UNCLASSIFIED ISI/RS-83-10 MDA903-81-C-0335 F/G 9/2 NL

END
DATE
FILMED
1-84
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A135 807

ISI Reprint Series
ISI/RS-83-10
November 1983

(13)

University of Southern California

Paul V. Mockapetris

Analysis of Reliable Multicast Algorithms for Local Networks

DTIC FILE COPY

INFORMATION SCIENCES INSTITUTE
213/822-1511
4676 Admiralty Way / Marina del Rey / California 90292

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ISI/RS-83-10	2. GOVT ACCESSION NO. AD-A135807	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Analysis of Reliable Multicast Algorithms for Local Networks		5. TYPE OF REPORT & PERIOD COVERED Research Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Paul V. Mockapetris		8. CONTRACT OR GRANT NUMBER(s) MDA903 81 C 0335
9. PERFORMING ORGANIZATION NAME AND ADDRESS USC/Information Sciences Institute 4676 Admiralty Way Marina del Rey, CA 90292		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ....
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE November 1983
		13. NUMBER OF PAGES 14
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document is approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

....

18. SUPPLEMENTARY NOTES

This report is a reprint of a paper that appears in the proceedings of the Eighth Data Communications Symposium, held in October 1983, in North Falmouth, MA. The Symposium is jointly sponsored by the Association for Computing Machinery Special Interest Group on Data Communications (SIGCOMM), the IEEE Computer Society Technical Committee on Computer Communications, and the IEEE Communications Society Technical Committee on Data Communication Systems, Computer Communications.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

broadcast, communication protocols, local networks, multicast, protocols, reliable broadcast, reliable multicast

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Local networks offer unique opportunities for supporting multicast transmissions. This paper describes and analyzes several families of multicast algorithms for local networks. The algorithms examined provide reliable service by dealing with the effects of transmission errors.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ISI Reprint Series
ISI/RS-83-10
November 1983

University of Southern California

Paul V. Mockapetris

Analysis of Reliable Multicast Algorithms for Local Networks

DTIC COPY INSPECTED 3

A-1

INFORMATION SCIENCES INSTITUTE

213/822-1511

4676 Admiralty Way / Marina del Rey / California 90292

This research is supported by the Defense Advanced Research Projects Agency under Contract No. MDA903 81 C 0335. Views and conclusions contained in this report are the author's and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

ISI Reprint Series

This report is one in a series of reprints of articles written by ISI research staff and published in professional journals and conference proceedings. For a complete list of ISI reports, write to

Document Distribution
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
USA

Contents

Introduction

Local networks use transmission media that make all transmissions equally available to all network interfaces; point-to-point communication is simulated by having all network interfaces but the one at the desired destination discard the transmission. These systems have the potential capability of distributing a single transmission to many destinations at the same cost as a point-to-point transmission. This raw broadcast capability is used sparingly by existing networks because of the cost and difficulty of transforming the raw capability into services which are compatible with existing high level protocols and appropriate to processes' needs. This paper compares algorithms for implementing reliable broadcast transmissions (one to all) as well as reliable multicast transmissions (one to a selected subset of all) in the context of existing bus and ring local networks, and also discusses modifications to link level hardware which improve multicast performance.

Interprocess communication need not be restricted to a single destination; a source process often needs to distribute the same information to several destination processes. When the target processes are distributed among local network hosts, we would like to use the broadcast capability of the medium to improve performance. Two forms of this type of communication are broadcast, in which a given transmission goes to all destinations, and multicast, in which information is sent to a specified subset of all destinations.

Because high level protocols allow every process to act as one or more destinations, broadcast has few applications at the process level; essentially no information is relevant to all processes in a network. Applications which use broadcast algorithms, such as routing update systems,[1] depend on addressing or some other mechanism to restrict the transmission. Thus broadcast services usually appear at a lower level of any system which supplies multicast services; should broadcast be needed at the higher level, it is treated as a special case of multicast. For example, network interfaces that recognize multicast addresses usually are based on a broadcast medium; interfaces that recognize a single broadcast address rely on host software to convert the broadcasts to multicast.

Whether implemented implicitly or explicitly, multicast transmissions are inherent in any application which includes a distributed database.[2,3,4,5,6] Multicast transmissions can search for an object among a set of server processes or inform all concerned processes of an event. Multicast queries enable multiple database servers to process queries in parallel. Multicast transmissions are useful in the update process; multicast allows for rapid update of redundant copies as well as rapid distribution of "ballots" in voting systems.

These goals imply the metrics which should be used to evaluate the effectiveness of any multicast system:

Speed Multicast services should be able to provide significantly faster service than the equivalent set of one-to-one transmissions. There are two times of interest. The *distribution* time is the time it takes for all multicast set members to receive a particular distribution. This time is especially important if the purpose is to allow parallelism. The *completion* time is the time it takes for the sender to know that all destinations have reliably received the distribution. This time is of special interest in update operations, and is also important in queries or searches when failure is detected as an absence of responses to a query. Both times can be measured in the number of transmissions necessary to achieve the corresponding result.

Cost Multicast operation have two types of cost: bandwidth on the medium and computation in the hosts. The bandwidth on the medium can be expressed as a transmission count. Host loading is measured in terms of the number of packet processing events (i.e. packet reception or transmission) that occur in the involved hosts. In a one-to-one transmission, the packet count is one, and the packet events are two, one for the sender and one for the receiver. With multicast, a given transmission will cost at least a transmission by the sender and receptions by every member in the multicast set.

Compatibility Multicast transmissions should follow the form and usage of one-to-one transmissions where possible. At the process level, the main difficulty is in reporting the result of a transmission request; rather than a simple success or failure, a multicast transmission may encounter success with some members of the multicast set and failure with others. Parallel difficulties arise with protocol definitions and mechanisms for sequencing, flow control, connection management, etc.

Reliability Some applications do not require absolute reliability. For example, a voting system may only need to generate a majority, and hence could use a slightly unreliable multicast facility. However, the majority of update and search applications are best served by reliable multicast. Even systems that have higher level error correction can benefit by avoiding the cost and delays associated with higher-level recovery mechanisms when databases are resynchronized.

Set definition Existing mechanisms for defining the processes that belong in a particular multicast group are not very general. For example, the Ethernet[7] uses a special address for each multicast set; each host that contains processes in multicast sets must recognize the set address(es) as well as the unique interface address. The network interfaces designed for the DCS system[8] contained special hardware to allow wildcarding of selected components of a segmented address. In both cases, address slots in the interface represent a limited resource that must be managed. Both systems rely on well known multicast set addresses, and hence do not allow dynamic creation of new sets.

Several proposals have been made for improving this type of mechanism. Rowe,[9] Pardo,[10] and others have proposed more sophisticated address recognition schemes that would allow multicast addresses to contain the equivalent of a function. For example, a multicast address might contain a component similar to a conventional multicast address which identifies a set of eligible processes (e.g. all database servers for employee records), as well as an application-specific predicate component (e.g. which are inverted by age). Both components would have a standard format. The first component would be a well-known value; the second component would have a standard interpretation but process specific values. For example, in the database case mentioned above, there would be a bit in the second component corresponding to each type of inversion kept in the databases. Boolean functions[9,10] and regular expressions[11] have been proposed for the evaluation predicate.

Set definition mechanisms must often provide other parts of the multicast support with information regarding the multicast set's population. For example, systems which collect separate acknowledgment transmissions (ACKs) from each member

of the multicast set must know how many members are currently in the set. If the predicate systems are used, sequence numbers must be updated in both members of the set that pass the predicate test as well as those that do not. Population information must track arrivals and departures from the multicast set. Ideally, we would like a scheme where the sender doesn't need this type of information unless required by the application (i.e. determining the number of processes that constitute a majority.)

Further work is needed in this area, and is beyond the scope of this paper.

Multicast implementations

Reliable multicast transmission can be decomposed into separate actions:

1. Assignment of a multicast set address.

2. A distribution transmission, and possibly retransmissions, that place the information to be multicast onto the medium.

3. Reception of the distribution transmission in the target hosts, followed by processing to discard duplicates and to route the information to target processes.

4. Generation and transmission of acknowledgments from receivers to the sending host.

5. Acknowledgment processing at the sending host.

This model illustrates the main performance issues in multicast.

The first of these is optimizing the performance of packet primitives in the network interface. Our goal is to optimize the multicast potential of the medium without incurring excessive cost in terms of processing events in the receivers of the distribution. This goal is achieved through measures to improve the probability that transmissions are successful and measures to rapidly discard irrelevant or duplicate transmissions. In this regard, multicast is more sensitive to the effects of errors than one-to-one transmission because although a failure may still double the cost, the cost of multicast increases with the size of the multicast set.

We also want to optimize the acknowledgment algorithm. In multicast, there is more distinct acknowledgment information than data to be acknowledged; hence special acknowledgment algorithms may be justified.

In the next two sections we examine these problems, and then compare the performance of several combinations of packet processing primitives and acknowledgment algorithms.

Packet primitives strategies and costs

Several techniques for improving interface performance are already in use in various systems. Interfaces should recognize multicast addresses instead of a single broadcast address; thus hosts that are not in the multicast set won't have to expend time to discard extraneous packets. Network interfaces can minimize packet loss through full duplex operation and by buffering strategies that allow reception of back-to-back packets from the medium. A fairly simple extension to this scheme

would be to reserve a buffer for each multicast connection so that multicast distributions would never be discarded due to lack of resources.

In situations such as saturation, where the ACK reliability isn't a problem, but the distribution may still need retransmission, duplicate detection can be made automatic by using two multicast addresses and a variation on the alternating bit protocol.[12] In this scheme, called *parity*, the sender uses one address for the initial transmission *and all retransmissions* of a message, and then switches to the other address for the next message. The receivers change addresses whenever they successfully copy a new message. Receivers that miss a message stay with the old address and eventually receive a retransmission; receivers that copy a message are spared receiving any future retransmissions. The parity system requires restrictions on the packet lifetime and outstanding messages that are rarely a problem in a local network.

The ultimate in performance is achieved by a network interface that performs duplicate detection, ACK generation, and ACK reception without host intervention. These interfaces are referred to as *filter* interfaces in further discussion. The parity scheme is a primitive example of automatic duplicate detection; various interfaces, such as the Hyperchannel[13] and others[14] incorporate automatic ACK generation, though at a low level in the protocol hierarchy. A scheme for a high-level version is described by Mockapetris.[11] These acknowledgments are transmitted immediately following the distribution they acknowledge, and hence are called *prompt* ACKs.

Acknowledgment strategies and costs

The focus for acknowledgments is eliminating the cost of ACK transmissions, either by moving the cost into the network interface or by reducing the number of ACKs required. Four types of multicast algorithms are considered:

1. Simulation algorithms, which achieve the effect of multicast using separate one-to-one transmissions. These algorithms are primarily of interest as a baseline for comparison.

2. Multiple acknowledgment algorithms, which distribute the multicast text using some sort of one-to-many transmission but collect ACKs via one-to-one transmissions. These algorithms are representative of today's systems.

3. Saturation algorithms, which rely on statistical arguments to avoid the need for acknowledgments.

4. Negative acknowledgment (NACK) algorithms, which require filtering and generate transmissions only when a transfer fails.

Simulation algorithms

The most straightforward way for a network which lacks any sort of one-to-many transmission capability to simulate multicast is for the sender to transmit separate messages to each destination and receive separate ACKs in return. Each destination requires two transmissions, so that a multicast set of N destinations requires 2N transmissions. Each transmission is created and received by a host, so a grand total of 4N packets are processed by all hosts. This method is usually the most expensive, but can be used with any medium or system of protocols. It requires all senders to maintain lists of multicast set members. Assuming that the receivers transmit ACKs as soon as they receive the

distribution, approximately 2N-1 transmissions will take place before all receivers have the distribution. Lost packets, whether distributions or ACKs, will result in 2 additional transmissions and 4 packet events.

If all hosts have filter interfaces, prompt ACKs replace host-generated ACKs and save 50 percent of both metrics.

An alternate method is to use a software ring of destinations: the source transmits the message to the first destination, which forwards the message to the next destination, etc. The last destination returns the message or an ACK to the sender. If intermediate ACKs are not returned, a total of N+1 transmissions are required. The advantages of this scheme are the reduction in traffic, and that the sender needn't maintain a list of all destinations; each member need only remember the next member. Each member of the set can also add new members. The drawbacks of this scheme are its slowness, since all transmissions are made in series, and its unfavorable performance in the presense of transient errors or host failures. When a packet is lost, the entire cost is doubled. No performance advantage is gained by using filter interfaces in this scheme.

Separate acknowledgment algorithms

The archetypes of multicast algorithms for a local network rely on various types of one-to-many distribution followed by one-to-one ACKs. The message is distributed in one transmission; N ACKs are subsequently returned. The host event cost includes the distribution transmission and N receptions; acknowledgments generate 2N events unless filter interfaces process the ACKs. In general, the cost of acknowledgments is greater than the cost of the message distribution.

The main variability in cost is due to the different number of hosts which may receive the one-to-many message. In a network with C hosts, a message addressed to the broadcast address is seen by all C hosts; if a multicast address is available, only the N destinations see the message.

These algorithms can be used with either a bus or a ring. However, the controlled access provided by a token system avoids the likely collisions between ACKs for a bus system. In systems where bus length reduces the effectiveness of carrier sense, some collisions are inevitable unless special pains are taken to spread out the ACKs; in the simplest CSMA system, these ACKs will often be precisely synchronized by the distribution transmission or other network traffic.

Saturation algorithms

Saturation algorithms do not use ACKs; instead they make a statistical argument about the probability of transmitting at least one copy of the message to every member of the multicast set. The basic principle is to transmit enough copies of the message to insure that at least one copy gets through to every destination.

If E is the probability that one or more destinations will not get the multicast (i.e., the desired overall error rate of multicast transmissions), and F is the the probability that a single network interface will tail to copy a multicast transmission (i.e. the link level error rate), then we wish to solve for M, the number of transmissions which are required to send to a set of N hosts with probability E of failure.

The probability of success for each host is

$$1-F^{M}$$

hence for all N hosts

$$1-E = (1-F^{M})^{N}$$

solving for M

$$M = \frac{\log(1-(1-E)^{1/N})}{\log(F)} \tag{1}$$

Figure 1 graphs M rounded to the next higher integer for several combinations of E and F.

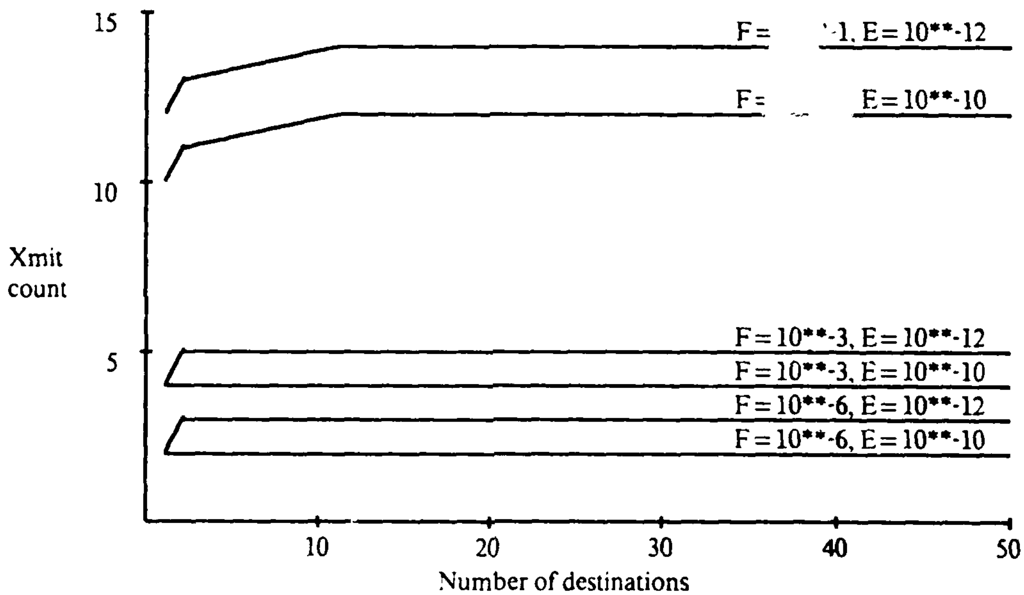


Figure 1: Multicast saturation transmission counts

For N = 1, M is equal to log(F)/log(E), that is, the ratio of the exponents of the error rates. M grows very slowly as N increases. Intuitively, this is because each additional destination benefits from the transmissions required by the other destinations. Contemporary network designs typically strive for an error rate E of 10^{-10} to 10^{-12}, although few empirical measurements have been made. Hence choosing a value for E is not a problem. The problem is determining the correct value for F, the interface-to-interface probability of packet loss. F is affected by several factors:

1. Errors due to line noise, clock-recovery errors, transmitter-receiver errors, and other noise-related problems in the hardware.

2. Transmissions which are discarded due to half-duplex interfaces.

3. Transmissions which are discarded due to interfaces which have a "recovery" time following receipt of a message before another message can be received.

4. Transmissions which are discarded due to lack of buffer space in the interface or host.

The noise component is fairly well understood. Shoch[15] reports that measurements on the Ethernet showed that a poorly designed interface had an error rate of 10^{-3}, whereas a well-designed interface had an error rate of less than 10^{-6}. The point-to-point communication links used in a ring do even better.

The remaining components are difficult to quantify because they are dependent on host and network load, and the retransmission policy followed by the sender. The preceding analysis depends on the fact that the transmissions are independent trials with a constant probability of success; even if this is a good assumption most of the time in real systems, it is almost impossible to know F exactly. Filter interfaces have a real advantage since they can eliminate these causes of packet loss. Conventional networks can still use saturation algorithms, but only at an artificially high value of F.

The packet events cost is very sensitive to the interface capabilities. A saturation algorithm used with a broadcast interface results in every host on the network having to process M packets; similarly a multicast address has every member of the multicast set processing M packets. In both cases, approximately M-1 of these packets are of no value and are discarded. A network of filtering interfaces is dramatically better because the duplicates are discarded by the interface.

These algorithms are also attractive in situations where the multicast set is large; in situations where collection of ACKs can be difficult; in situations where quick delivery is more important than host processing costs; and in situations where F is well known. Because this algorithm assumes that errors will occur, its behavior is the same whether or not errors occur.

Negative acknowledgment algorithms

The strength of the saturation algorithm is that it avoids the need for N ACKs; its primary weakness is its inability to deal with interfaces which are unable to receive due to buffering problems and other load-induced conditions which may persist for indeterminate time. The independent trials assumption at the heart of the theoretical model is very difficult to realize in practice.

This problem is avoided by negative acknowledgment (NACK) algorithms, which assume filter interfaces. Members of the multicast set which receive the message correctly don't transmit a prompt ACK; member of the multicast set which would like to be able to copy the message, but cannot, send a prompt NACK. The NACK demands a retransmission by the sender of the distribution message. Because NACK generation requires no resources, interfaces can always generate NACKs. Thus load-induced conditions are removed from estimates of F, and F is driven by the noise-related causes which presumably are random.

NACK algorithms relate to separate acknowledgment algorithms in that the termination conditions are equivalent under DeMorgan's law. A separate acknowledgment system terminates on the basis of a logical AND condition over the individual ACKs; the NACK algorithm relies on the logical NOR of NACK transmissions. The functional difference is that the NOR predicate can be deduced from any single destination which sends a NACK; the AND requires all results. In use this means that the NACK protocol doesn't need to know how many destinations are in the multicast set; this is somewhat of a disadvantage in that it cannot detect destination failures that lose state, such as a host crash and restart.

In a bus system, multiple destinations may wish to send a NACK, and these transmissions will collide. However, the presence or absence of a NACK is all the information which is required. Hence NACKs could use the same mechanism that is used to achieve collision consensus enforcement; the collision is equivalent to a NACK. In order to guarantee an acceptable level of reliability, the sender should retransmit until M transmissions have *not* resulted in NACKs (collisions); these extra transmissions are required for bit errors in the receivers which are not due to causes that the sender can detect.

In a ring network, the NACK system can be particularly reliable and effective. The sender transmits the distribution message followed by a "blank" message. If both return unchanged to the sender, then the multicast is complete. If any destination is unable to copy the distribution message, it overlays the blank message with a NACK. This scheme is superior to the *match/accept* bit scheme used in the RI[16] and LNI[8] in that it neither requires that the interfaces examine bits of media data before changing them nor is it sensitive to a small number of bit errors. Instead, the destinations simply emit a prompt NACK and its associated CRC.

The logical extension to this ring scheme is to encode whether the message is a multicast or not as a bit in the message; the interface hardware could invert the sense of prompt ACKs automatically.

Algorithm comparisons

Table 1 summarizes the algorithm comparisons. The distribution time, completion time, and packet events columns assume that no packets are lost. The "per error events" column is the maximum cost of recovering from loss of one packet.

Any system that uses a multicast distribution channel has equivalent best case distribution time. However, when errors occur, the distribution time is related to the completion time because some estimate of the completion time is presumably included in the retransmission interval. Hence if it is important to bound the distribution time even in the presence of errors, NACK and saturation algorithms offer the best performance.

In systems using existing technology, the two most practical choices are multicast with separate ACKs and saturation using parity. In terms of completion time, the saturation algorithm is equivalent in cost at $N = 4$ and better for larger multicast sets; each additional member saves one message. In terms of packet events, saturation is equivalent at $N = 2$ and grows better at a rate of $2N$; if the cost of error correction is considered, saturation is always better for any multicast. Saturation also has the advantage that the sender need not keep track of the current membership in the multicast set; given M's insensitivity to N, the sender does not even have to track the size of the multicast set. The only disadvantage to saturation is the difficulty in including load-induced packet loss into the basic channel error rate. As VLSI network chips become available that include buffer management, this problem should become quite tractable.

The algorithm comparison also suggests some directions for future development. The filtering concept doesn't improve the completion time unless it includes an inversion of acknowledgment strategy from ACKs to NACKs. Given the surprising success of saturation, it might be more worthwhile to concentrate on strategies for priority buffering so that the saturation estimate of F is improved.

Table 1: Best case multicast algorithm comparison

Major algorithm	Option used	distribution time	completion time	packet events	per error events
Simulate	one by one	2N-1	2N	4N	4
	filter ACKs	2N-1	2N	N + 1	1
	software ring	N	N + 1	2(N + 1)	2(N + 1)
Separate ACKs	broadcast address	1	N + 1	C + 2N + 1	C + 2N + 1
	multicast address	1	N + 1	3N + 1	3N + 1
	filter ACKs	1	N + 1	N + 1	2
Saturate	broadcast address	1	M	M(C + 1)	0
	multicast address	1	M	M(N + 1)	0
	filter/parity	1	M	N + M	0
NACK	ring	1	1	N + 1	N + 1
	Ethernet	1	M	N + 1	0

Legend:

N = number of destinations desired C = number of hosts on network

M = result of saturation formula (1); assuming single message error rate of 10^{-3} and desired net error rate of 10^{-12}, M = 5

References

1. McQuillan, J.M., Richer, I., and Rosen, E.C., "The New Routing Algorithm for the ARPANET," IEEE Transactions on Communications, Vol. com-28, no. 5, May 1980.

2. Gifford, D.K., "Weighted Voting for Replicated Data," Proceedings Seventh Symposium on Operating Systems Principles, December 1979.

3. Gifford, D.K., "Violet, an Experimental Decentralized System," XEROX PARC Technical Report CSL-79-12, September 1979.

4. Rothnie, J.B., Goodman, N., and Bernstein, P.A., "The Redundant Update Methodology of SDD-1: A System for Distributed Databases (The Fully Redundant Case)," report no. CCA-77-02, Computer Corporation of America, 1977.

5. Stonebreaker, M., "Concurrency Control and Consistency of Multiple Copies of Data in Distributed INGRESS," IEEE Transactions on Software Engineering, May 1979, pp. 188-194.

6. Thomas, R.H., "A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases," ACM Transactions on Database Systems, June 1979, pp. 180-209.

7. Xerox Corp., Intel Corp., and Digital Equipment Corp., "The Ethernet, A Local Area Network, Data Link Layer and Physical Layer Specifications," Version 1.0, September 1980.

8. Mockapetris, P.V., Lyle, M.R., and Farber, D.J., "On the Design of Local Network Interfaces," IFIP Congress, August 1977, pp. 427-430.

9. Rowe, L.A., Birman, K.P., "Network Support for a Distributed Data Base System," Proceedings Fourth Berkeley Conference on Distributed Data Bases and Computer Networks, 1979.

10. Pardo, R., and Liu, M.T.,"Multidestination protocols for distributed systems," Proceedings of the Computer Networking Symposium, National Bureau of Standards, Gaithersburg, Maryland, 1979.

11. Mockapetris, P., "Communication Environments for Local Networks," USC/Information Sciences Institute, ISI/RR-82-103, December 1982.

12. Bartlett, K.A., Scantlebury, R.A., and Wilkenson, P.T., "A Note on Reliable Full-Duplex Transmission over Half-Duplex Links," CACM, vol. 12, no. 5, May 1969.

13. Thornton, James E., "Overview of the HYPERchannel," Proceedings COMPCON, 1979.

14. Tokoro, M., and Tamaru, K., "Acknowledging ETHERNET," Proceedings COMPCON, Fall 1977.

15. Shoch, J.F., and Hupp, J.A., "Performance of an Ethernet Local Network: A Preliminary Report," Proceedings of the local area communication network symposium, May 1979, pp. 113-125.

16. Mockapetris, P., D.J. Farber. "Experiences with The Distributed Computer System," Technical Report 116, Department of Information and Computer Science, University of California, Irvine.